Introduction

My project revolved around developing a DNN to generate captions on images. I originally planned to implement a pre-trained CNN like ResNet as an Encoder and train a transformer decoder from scratch. As mentioned in class, training language decoders is computationally expensive and requires a lot of time, electricity, and computing resources.

After experimenting with the transformer architecture and building my own Encoder Decoder, I implemented a pre-trained transformer from HuggingFace.

Since implementing a pre-trained model did not require the same level of debugging, I spent a large amount of my time reading papers from Arxiv to get a better idea of the SOTA DL implementations for NLP and Computer Vision.

Dataset description

My pre-trained model was trained on the COCO MS dataset.

Baseline approach description

As mentioned above, I originally intended to use ResNet as an encoder and connect the encoded state to the unmasked portion of a transformer decoder. I then intended to use teacher forcing to train the ResNet-transformer and seed each training sample with a token on input and append an token to the output.

Unfortunately, because of transformers must be structured differently during training and testing (transformers do not use input to the decoder during testing) I was unable to train my model. This is an aspect of the code that I will likely look into over the summer for my own personal growth.

Method description

Some of the works I read included Attention Is All You Need, the ViT paper, CPTR paper, and an online explanation of GPT2.

ViT explains how an image can be split up into 16x16 pixel patches and each patch can be thought of similarly to how a transformer processes words. Each patch carries some contextual information about the image depending on its location on the image. In the ViT paper, they use this transformer encoder and feed it to a dense layer in order to classify images.

CPTR attempts to connect a transformer image encoder similar to the one defined in ViT to a transformer decoder in order to generate text captions. Some of the improvements defined in the CPTR paper are using a GeLu activation function in the dense layers of the encoder/decoder blocks after multi-head

attention. This allows the model to drop out and zoneout neurons simply through the activation function, and simplify computations.

To my surprise, GPT2 is only a transformer decoder. This allows the model to generate text more efficiently than it would by using an encoder decoder architecture which was designed for sequence to sequence translation.

Evaluation

Had I trained my own model on the COCO MS dataset, I would have been able to use a BLEU score to evaluate my output on the expected text output. Without the COCO

Discussion

While I was implementing the ViT-GPT2 model through HuggingFace, I had some ideas for areas of future study. With luck, I would like to research these and/or implement these ideas over the next few years.

1. Image2Vec

I have read a bit about the Word2Vec block and GLoVE which encode a word through learned representations of a CBOW representation. To my understanding, Word2Vec tries to predict a valid context word given a center word. The decoder is thrown away after training and the W2V encoder is used to encode words into a meaningful feature space.

I would like to experiment with applying a similar algorithm to image patches for better image captions, classifications, and bounding boxes for computer vision. I wonder if there is an Image2Vec model which could generate useful image encodings by attempting a similar problem of predicting a "context patch" from a "center patch".

2. Convolutional or Multiheaded Word Encoding

Word2Vec implements an encoder decoder to learn meaningful word representations but generally does this with one layer. I wonder if implementing a similar encoder decoder with a CNN or implementing a multiheaded Word2Vec block would yield a better word representation than before.

Conversely, maybe this is not necessary for effective representations.

Conclusion

This project has taught me about using transformer to solve language related problems, but more importantly, I learned the importance of reading related work in one's field. I am starting to realize that it is easy to become so immersed

in one's own work that he becomes blind to all the important contributions around him.

One part of the class that really stood out to me was when you did a demo of Arxiv and showed that there were 70+ papers published on the website that day; each of which consisted of the work of a group of researchers for 6+ months.

I hope to be more diligent about reading related works.

Appendix (contribution of each team member)

I was the only team member in my group.